Real-Time Linux with *PREEMPT_RT* training

Course duration

3 half days – 12 hours

Language

Materials	English
Oral Lecture	English French

Trainer

One of the following engineers

- Maxime Chevallier

Contact

training@bootlin.com

+33 484 258 097

Online seminar

Audience

Companies and engineers interested in writing and benchmarking real-time applications and drivers on an embedded Linux system.

Training objectives

- Be able to understand the characteristics of a real-time operating system
- Be able to download, build and use the *PREEMPT_RT* patch
- Be able to identify and benchmark the hardware platform in terms of real-time characteristics
- Be able to configure the Linux kernel for deterministic behavior.
- Be able to develop, trace and debug real-time user-space Linux applications.

Prerequisites

- **Knowledge and practice of UNIX or GNU/Linux commands**: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides.
- **Minimal experience in embedded Linux development**: participants should have a minimal understanding of the architecture of embedded Linux systems: role of the Linux kernel vs. user-space, development of Linux user-space applications in C. Following Bootlin's Embedded Linux course allows to fulfill this pre-requisite.
- **Minimal English language level: B1**, according to the *Common European Framework of References for Languages*, for our sessions in English. See the CEFR grid for self-evaluation.

Pedagogics

- Lectures delivered by the trainer, over video-conference. Participants can ask questions at any time.
- Practical demonstrations done by the trainer, based on practical labs, over video-conference. Participants can ask questions at any time. Optionally, participants who have access to the hardware accessories can reproduce the practical labs by themselves.
- Instant messaging for questions between sessions (replies under 24h, outside of week-ends and bank holidays).
- Electronic copies of presentations, lab instructions and data files. They are freely available here.

Certificate

Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.

Disabilities

Participants with disabilities who have special needs are invited to contact us at *training@bootlin.com* to discuss adaptations to the training course.

bootlin

bootlin.com

Required equipement

Mandatory equipment:

- Computer with the operating system of your choice, with the Google Chrome or Chromium browser for videoconferencing.
- Webcam and microphone (preferably from an audio headset).
- High speed access to the Internet.

Optionnally, if the participants want to be able to reproduce the practical labs by themselves, they must separately purchase the hardware platform and accessories, and must have a PC computer with a native installation of Ubuntu Linux 24.04.

Hardware platform for practical labs

STM32MP1 Discovery Kit

One of these Discovery Kits from STMicroelectronics: **STM32MP157A-DK1**, **STM32MP157D-DK1**, **STM32MP157C-DK2** or **STM32MP157F-DK2**

- STM32MP157, dual Cortex-A7 processor from STMicroelectronics
- USB powered
- 512 MB DDR3L RAM
- Gigabit Ethernet port
- 4 USB 2.0 host ports
- 1 USB-C OTG port
- 1 Micro SD slot
- On-board ST-LINK/V2-1 debugger
- Arduino compatible headers
- Audio codec, buttons, LEDs
- LCD touchscreen (DK2 kits only)

Training Schedule

Half day 1

Lecture	Introduction to Real-Time behaviour and determinism	▪ Definition of a Real-Time Operating System ▪ Specificities of multi-task systems ▪ Common locking and prioritizing patterns ▪ Overview of existing Real-Time Operating Systems ▪ Approaches to bring Real-Time capabilities to Linux
Lecture	The *PREEMPT_RT* patch	▪ History and future of the *PREEMPT_RT* patch ▪ Real-Time improvements from *PREEMPT_RT* in mainline Linux ▪ The internals of *PREEMPT_RT* ▪ Interrupt handling: threaded interrupts, softirqs ▪ Locking primitives: mutexes and spinlocks, sleeping spinlocks ▪ Preemption models
Demo	Building a mainline Linux Kernel with the *PREEMPT_RT* patch	▪ Downloading the Linux Kernel, and applying the patch ▪ Configuring the Kernel ▪ Booting the Kernel on the target hardware
Lecture	Hardware configuration and limitations for Real-Time	▪ Interrupts and deep firmware ▪ Interaction with power management features: CPU frequency scaling and sleep states ▪ DMA

Half day 2

Lecture	Tools: Benchmarking, Stressing and Analyzing	▪ Benchmarking with *cyclictest* ▪ System stressing with *stress-ng* and *hackbench* ▪ The Linux Kernel tracing infrastructure ▪ Latency and scheduling analysis with *ftrace*, *kernelshark* or *LTTng*
Demo	Tools: Benchmarking, Stressing and Analyzing	▪ Usage of benchmarking and stress tools ▪ Common benchmarking techniques ▪ Benchmarking and configuring the hardware platform
Lecture	Kernel infrastructures and configuration	▪ Good practices when writing Linux kernel drivers ▪ Scheduling policies and priorities: *SCHED_FIFO*, *SCHED_RR*, *SCHED_DEADLINE* ▪ CPU and IRQ Affinity ▪ Memory management ▪ CPU isolation with *isolcpus*

Half day 3

Lecture	Real-Time Applications programming patterns	▪ POSIX real-time API ▪ Thread management and configuration ▪ Memory management: memory allocation and memory locking, stack ▪ Locking patterns: mutexes, priority inheritance ▪ Inter-Process Communication ▪ Signaling
Demo	Debugging a demo application	▪ Make a demo userspace application deterministic ▪ Use the tracing infrastructure to identify the cause of a latency ▪ Learn how to use the POSIX API to manage threads, locking and memory ▪ Learn how to use the CPU affinities and configure the scheduling policy